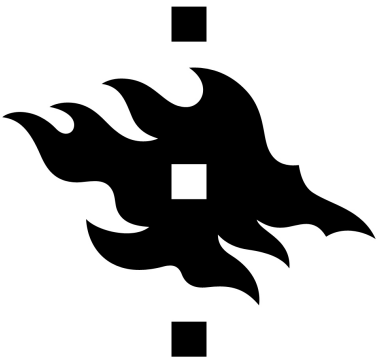

UNIVERSITY OF HELSINKI

Patch-based image representation and restoration

by

Markus Juvonen

Submitted for the degree of Master of Science
Department of Mathematics and Statistics
December 2016

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta/Osasto — Fakultet/Sektion — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Mathematics and Statistics

Tekijä — Författare — Author
Markus Juvonen

Työn nimi — Arbetets titel — Title

Patch-based image representation and restoration

Oppiaine — Läroämne — Subject
Mathematics

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's thesis	December 2016	41 p.

Tiivistelmä — Referat — Abstract

This thesis strives to familiarize the ideas behind the success of patch-based image representations in image processing applications in recent years. Furthermore we show how to restore images using the idea of patch-based dictionary learning and the k-means clustering algorithm.

In chapter 1 we introduce the notion of patch-based image processing and take a look at why dictionary learning using sparsity is a hot topic and useful in processing natural images. The second chapter aims to formulate the different methods and approaches used in this thesis mathematically. Dictionary learning, the k-means algorithm and the Structural similarity index (SSIM) are in the main focus. Chapter 3 goes into the details of the experiments. We present and discuss the results as well. The fourth and final chapter summarizes the main ideas of the thesis and introduces development suggestions for further investigation based on the methods used.

Using a fairly simplistic patch-based image processing method we manage to reconstruct images from a set of similar images to a reasonable extent. As the main result we see how the size of the patches as well as the size of the learned dictionary effects the quality of the restored image. We also detect the limitations and problems of this approach such as the appearance of patch artifacts which is an issue to attack and resolve in following studies.

Avainsanat — Nyckelord — Keywords
Image processing, Dictionary learning, K-means clustering, Denoising, Structural similarity

Säilytyspaikka — Förvaringsställe — Where deposited
Electronic archive

Muita tietoja — Övriga uppgifter — Additional information

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta/Osasto — Fakultet/Sektion — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Matematiikan ja tilastotieteen laitos

Tekijä — Författare — Author
Markus Juvonen

Työn nimi — Arbetets titel — Title

Valokuvien tilkkupohjainen esitys ja käsittely

Oppiaine — Läroämne — Subject		
Matematiikka		
Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Pro gradu -tutkielma	Joulukuu 2016	41 s.

Tiivistelmä — Referat — Abstract

Tämän työn pyrkimys on tutustuttaa lukija tilkkupohjaisen esitystavan takana oleviin ajatuksiin. Lisäksi näytämme kuinka rekonstruoida valokuvia käyttämällä tilkkupohjaista kirjasto-oppimista k-means klusterointi algoritmin avulla.

Ensimmäisessä kappaleessa esittelemme tilkkupohjaisen esitystavan idean ja tarkastelemme harvan kirjasto-oppimisen hyödyllisyyttä luonnollisten kuvien käsittelyssä. Muotoilemme työssä käytetyt menetelmät matemaattisesti kappaleessa kaksi. Pääpaino on kirjasto-oppimisessa, k-means klusterointi algoritmissa sekä SSIM indeksissä, joka pyrkii kuvien rakenteellisen yhtäläisyyden arvioimiseen. Kappaleessa kolme käydään läpi työn kokeellinen osuus. Esittelemme tulokset ja lisäksi keskustelemme niistä samassa kappaleessa. Lopuksi vedämme yhteen tämän työn tärkeimmät ideat sekä annamme kehitysideoita mahdollisia jatkotutkimuksia varten kappaleessa neljä.

Käyttämällä melko pelkistettyä tilkkupohjaista kuvankäsittelymenetelmää onnistumme rekonstruoimaan valokuvan kokoelmasta samankaltaisia kuvia kohtuullisen hyvin. Tilkkujen sekä opitun kirjaston kokojen vaikutukset kuvan rekonstruktion laatuun ovat työn keskeisimpiä tuloksia. Havaitsemme myös käyttämämme lähestymistavan rajoitukset ja ongelmat, kuten tilkku-artefaktien ilmestymisen rekonstruoituun kuvaan. Tämän ongelman ratkaisemiseksi jatkotutkimukselle on tarvetta.

Avainsanat — Nyckelord — Keywords
Kuvankäsittely, kirjasto-oppiminen, klusterointi, kohinanpoisto, rakenteellinen yhtäläisyys

Säilytyspaikka — Förvaringsställe — Where deposited
Sähköinen arkisto

Muita tietoja — Övriga uppgifter — Additional information

Acknowledgements

I want to thank Professor Samuli Siltanen for supervising this work and for giving me the opportunity to incorporate my passion for photography into this thesis.

Contents

List of Figures

Chapter 1

Introduction

Imagine being able to reconstruct any arbitrary image out of pieces from other images. If you have an unlimited amount of these pieces and all the time in the world this sounds like a doable task. Also the smaller the available pieces are the more likely they are to fit perfectly into the image reconstruction. Having a set of similar images compared to the original available would probably save you some time finding suitable pieces. All together this task would not be much different than assembling a mosaic. In this thesis we will reconstruct digital images using smaller pieces taken from similar images. In image processing we usually know what kind of image we are trying to enhance so this prior knowledge is available and can be exploited. We will approach the restoration problem by using a common algorithm that helps us sort the pieces into groups of similar pieces. You might have assembled a jigsaw puzzle sometime and before just trying to connect each piece together with all the others, one at a time, you might want to look for similar pieces first. Pieces that make up the borders, pieces that have the same color or pieces with similar image structures printed on them. This will make it faster and easier to find the right pieces of the puzzle. In our case we will take a representative of each of these groups of pieces and build a dictionary that can be used to restore any given image.

We will also look into the possibility of restoring an image that has some additional noise in it by using the same method. An interesting question we will investigate is how many different groups of puzzle pieces we can find or need to find to get a good reconstruction. Also the effect of the size of the pieces will be examined.

The reader should be somewhat familiar with basic linear algebra to fully understand

the things we will discuss. Prior knowledge about digital imaging and signal processing will also help to understand the content of this thesis better as we won't go into details and basics about how a digital system camera works and how a digital image is formed. However the aim is that anyone will be able to follow the main ideas and implementations of this work without previous comprehension in this field. If the reader is not familiar with the basics of digital photography, or would like to know more about image processing in general, I recommend to take a look in some introductory books about these topics.

Before diving deeper into the more formal mathematical side of the image representation problems addressed in this thesis let's have a look why it makes sense to approach these problems by looking at smaller sub-images, so called image patches.

1.1 Patch-based Representation and the Structure of Natural Images

Think about a photo you have seen lately, or better yet get your mobile phone out and look at any picture you have taken lately and zoom into the image until you have reached an area of approximately 10×10 pixels. Depending on the part of the image you zoomed into you probably ended up with either a mostly uniform piece of the image(e.g. wall, sky, any out of focus background), a part of some repetitive pattern or texture (e.g. fabric, skin, wallpaper), or a part containing a clear edge between two different areas [1].

If you would now move just a little bit in any direction of the zoomed in image you would likely end up with a similar piece of the image, unless of course you are moving over an edge. This happens due to the fact that natural images are full off redundancy, meaning that we could discard some of the information but still be able to represent the same image well. This inherent property of natural images gives us a chance to find image representation models that only need a small amount of variables to represent the given image data. This principle of representing some phenomenon with as few variables as possible is generally defined as sparsity, sometimes also called parsimony [2].

If you did not find a suitable image to investigate yourself or have trouble imagining all of this, no problem, let's have a look at a concrete example to help you get the hang

of it. In Figure 1.1 we see three different looking 10×10 pixel sized pieces taken from an image of a Timberman beetle (*Acanthocinus aedilis*). The first piece starting from the top, shows part of the background that is quite evenly gray. The second one represents part of the beetles forehead that looks like it is covered with some fur and the last one is from the edge between one of the beetles legs and the background. We notice from the close-ups that these three parts of the image seem really different but at the same time we can easily find areas of the image where we would suspect pieces to look similar or even identical to one of these three. Taking advantage of this redundancy in images is one of the core ideas in many image processing applications.

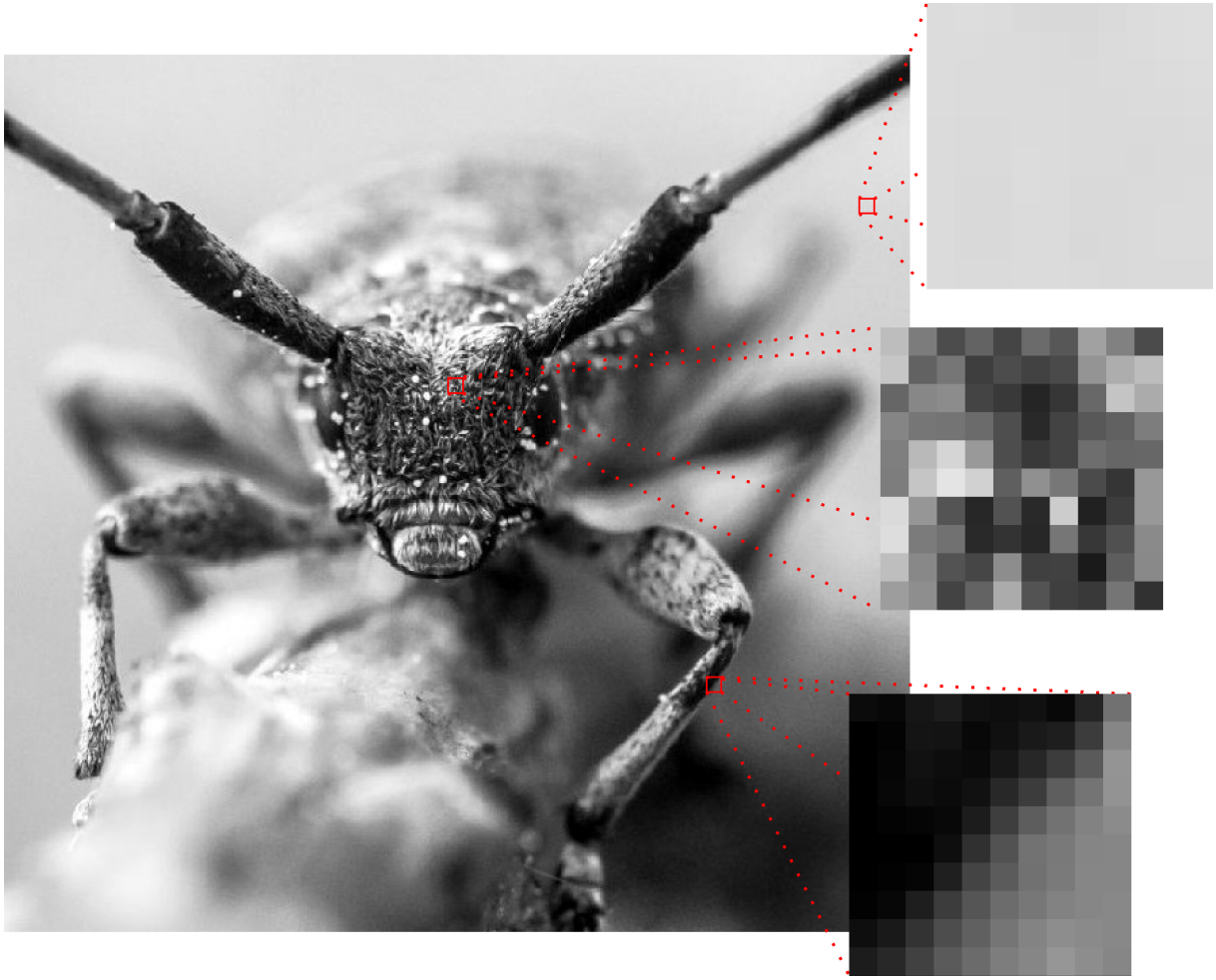

Figure 1.1: Zooming into three different 10×10 patches of a natural image.

We are going to call the small pieces of an image, that we zoomed into before, patches. In some of the related literature image patches are also called image blocks or windows, but since the term patch is starting to become the preferred term in the image processing

community we will be using that term throughout this thesis.

We will be working with square patches of various sizes from 4×4 pixels up to 16×16 pixels. The size of the patches used is often selected to be slightly bigger than the largest recognizable texture elements in the images to be processed. Preferably one could use different sized patches for different regions of images and there are some methods that use this approach but in our overview we will use only one fixed size of patches at once. By changing the patch sizes we will be able to see how this affects the quality of our restorations.

Using a patch-based approach in image processing simply means the image is divided into these small patches and each patch is then processed individually. After the processing steps the final output image is reconstructed out of the individually processed patches.

Working with image patches instead of working with the entire image is preferable in several ways. One reason to work with patches is the so called curse of dimensionality. The expression was made popular by Richard E. Bellman and it refers to the rapid increase of the entire space relative to the space occupied by data points as dimensions are added. This makes it much easier and more reliable to learn a certain representation model from smaller patches rather than the entire image. Also the computational demands of patch-based approaches are often significantly lower.

A downside of processing each patch individually is the appearance of patch/block artifacts in the resulting image (see Figure 1.2 for an example). There are ways of reducing the emergence of these artifacts and a common way is to use overlapping patches instead of non-overlapping patches in the image processing phase. One can also post-process the resulted images to reduce the "blocking effect" [3].

If we think about the jigsaw puzzle example again, these blocking artifacts can be thought of as the problem we face trying to fit a similar looking piece from another puzzle to pieces from different puzzles. The colors and structures might match but they won't fit smoothly together and interlock as hoped.

There are also a few advantages using non-overlapping blocks, compared to overlapping blocks. We fend off over-smoothing of the image textures when averaging over the overlapping regions, and it requires less computing time [4]. In this thesis we will only work with non-overlapping patches.

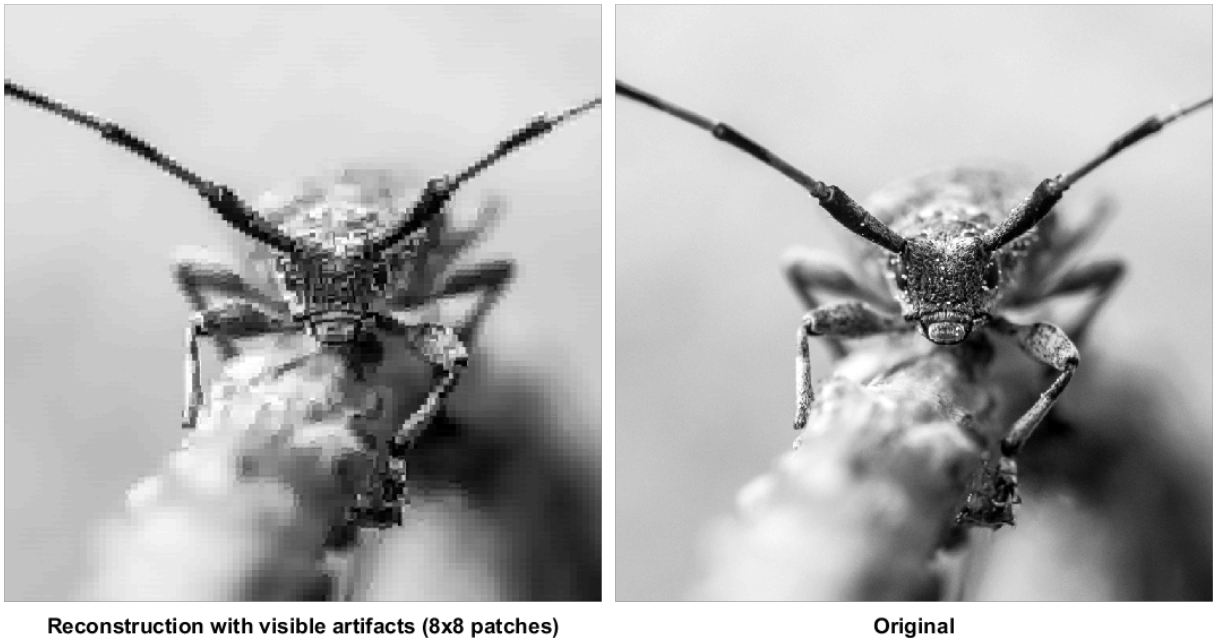

Figure 1.2: Vizualization of patch artifacts in a reconstruction with 8x8 patches compared to the original image.

Patch-based methods have truly been a hot research area in image processing in the last years. The amount of published research papers is huge and especially methods using natural images and their patches have brought significant improvements to previous approaches in many different image processing applications. Research in the past decade has in fact shown that algorithms created using patch-based methods have proven superior in image processing areas such as image denoising, inpainting and super-resolution [5]. A more recent step in image processing has been the combination of patch-based models with sparse representations. We will look more at sparsity in the next section as it is often closely linked with dictionary learning.

In this thesis we will work with so called natural images. But what is a natural image and how does it differ from unnatural ones? How to define a natural image? The most natural way (pun intended) that comes to mind is to include all images that are not artificially constructed nor images of such artificially generated scenes. By these artificially constructed images we mean "synthetic", computer generated images.

Hyvärinen et al [6] make the connection of natural images and the human visual system by defining natural images to be some set that we believe has similar statistical structure

to that which our visual system is adapted to. Their definition leads to a clear distinction between purely natural scenes and scenes that include the marks of human engineering, due to the fact that our visual system developed during a long period of time in which our environment was much different from todays.

We will include images of man-made objects (e.g. buildings, streets) and set-ups of natural scenes in our definition of natural images since it does not have a drastic impact on our forthcoming analysis and we would like to think of humans as part of the natural world. When you think about it, it is obvious that man-made scenes should include even more redundancy than natures random events. Take for example a forest in its natural state and compare it to a forest managed by humans or even more extremely a forest entirely planted by people, where trees often form straight lines and are of equal size and lack any variety of species. But as stated before we won't exclude these possible images from our definition of natural images.

So we have discussed the advantages of working with image patches. Next we want to glance upon a field that uses these patches to learn a so called dictionary. This dictionary consists of representatives of the jigsaw puzzle piece groups we talked about earlier. So every group will be represented by a single representative in this collection of representatives called dictionary.

1.2 About Dictionary Learning and Sparsity

In the past decade the search for the best possible dictionary has moved from trying to generate universal dictionaries that work for every data signal to constructing a dictionary that uses the data at hand or similar exemplars, allowing better adaptation to the data being processed/analyzed. This approach is called Dictionary learning and it has proven to provide better results in image processing applications than using the generic dictionaries. These predefined dictionaries such as wavelets, curvelets or contourlets have their advantages but are usually good for certain types of signals only. They might not perform as well in presenting more intricate patterns and features that are often present in natural images[7].

The key assumption behind the dictionary learning approach is that the structure of

complex natural occurrences can be more precisely obtained directly from the data than by using a mathematical characterization. We can use the knowledge of the nature of the data to be analyzed as a prior. A direct benefit of this is that a finer adaptation to specific details of the data becomes possible, allowing the replacement of universal models [8].

As the popularity of the learning approach has grown the amount of different proposed algorithms to learn a dictionary has increased immensely. The pioneering work in dictionary learning did not come from inside the image processing and computer vision community but from the field of neuroscience. Olshausen and Field developed a model that uses training data to learn an adapted dictionary while studying the mammalian visual cortex [9]. Their intention was to show that the structure of natural images is related to theories about the visual system of mammals. They came up with the idea to use image patches because the full images were too large for learning a dictionary matrix. Since then the dictionary learning approach has been adapted to numerous applications in image processing. One can read more about the history and development of different kinds of dictionaries for sparse representations in [8].

A learned dictionary is computationally more expensive than a fixed one but it has the advantage of being tailored to the characteristics of the desired solution. The learned dictionary should be robust to irrelevant features, and the number of training images should be large enough to ensure that all image features are represented [4]. Thus dictionaries are typically overcomplete (i.e redundant) meaning the amount of atoms in the dictionary is larger than the patch dimension, even though a choice of fever dictionary atoms than signal dimensions often leads to reasonable results in many applications [2].

Image representation models are often separated into two ways of using the dictionary to represent a signal, analysis and synthesis. We are more interested in the synthesis case as we aim to restore a signal as a linear combination of the dictionary atoms. In the overcomplete case we actually have an infinite amount of possibilities to represent the signal. This means we have a new task in looking for the best possible representation. A common notion is that the best dictionary is the one providing the sparsest representation [7]. The search for this sparse solution is conducted by introducing a cost function of some sort that promotes sparsity [8]. This step is generally referred to as sparse coding.

To learn our dictionary in this thesis, we will be using a basic and popular data clustering algorithm called k-means. In the Finnish language we would refer to this kind of simplistic model as a "karvalakkimalli", a so called "fur-cap model" that does not have all the latest advancements but it does the job you want it to do. K-means can't compete with more sophisticated dictionary learning algorithms like MOD [10] or K-SVD [11] but under the right conditions it does well for example in the area of feature-learning [12]. By using k-means will be able to have a look at some interesting properties of dictionary learning without having to rely on a difficult and complex algorithm.

In Chapter 2 we will formulate the problem at hand, and our means to solve the problem mathematically to give us a solid foundation for what we are trying to accomplish. Then in Chapter 3 we will go through the experiments and results and finally we discuss the results and introduce some development ideas in Chapter 4.

Chapter 2

Mathematical formulations

Before looking at our experiments we will have a more formal discussion on the mathematics behind the subject matter. As stated before at least a basic knowledge of linear algebra is needed to fully understand the mathematical concepts we are dealing with. The good thing about our way of solving the problems in this thesis is that we don't need that much heavy and difficult mathematical formulation to accomplish our objectives.

2.1 Patch-based Representation of an Image

We talked about the significance of patch-based approaches in signal and especially image processing in section 1.1. Let us now get into the details on how to divide a square image I with n^2 pixels into square patches of the size $Q = q \times q$ and how to represent the data. We work with square images and patches for the sake of simplicity but the use of different shaped patches would be just as possible. The number of patches we get is $R = \frac{n}{q} \times \frac{n}{q}$.

We can index the patches x with single integers $\{1, ..., R\}$ according to their location in the image. We will also vectorize the patches as vectors in $\mathbb{R}^Q$ so we can store them as columns of a matrix $X \in \mathbb{R}^{Q \times R}$.

Vectorization of an image patch means nothing more than performing a linear transformation which converts the patch matrix P into a column vector $vec(P)$ as in Figure 2.1. Vectorizing the patches makes it easier to perform all necessary computations in the future.

Let's look at an quick example. Figure 2.2 shows us an image that consists of $256 \times$

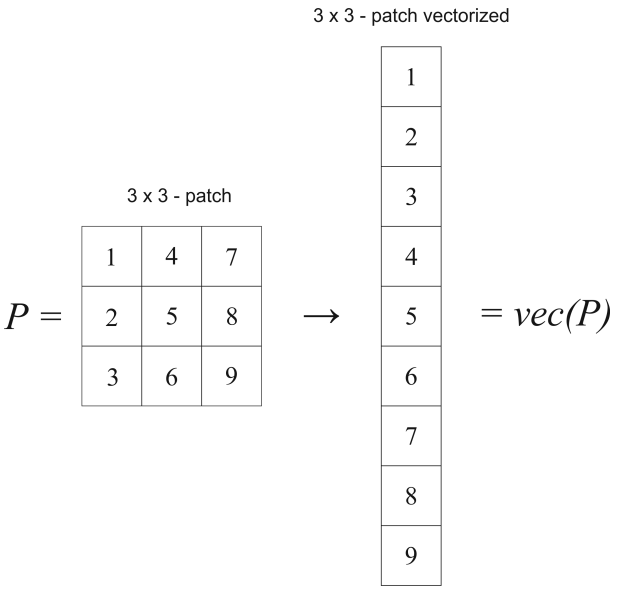

Figure 2.1: Example picture of the vectorization of a 3×3 image patch.

$256 = 65536$ pixels divided into 4×4 patches. The number of patches will be $\frac{256}{4} \times \frac{256}{4} = 4096$. Feel free to count the patches from the figure for reassurance.

2.2 Image restoration

The following formulations are based on [13, 14]. Reconstructing a high quality image from its degraded (e.g. blurred, noisy, downsampled) measurements has many relevant application fields, such as medical imaging, entertainment, remote sensing, etc. For an observed image Y, this image restoration problem can be formulated as

$$Y = HX + v \tag{2.1}$$

where H is a degradation operator matrix, X is the unknown, original image vector and v is the additive noise vector. With different matrices H this equation represents different image restoration problems. For example if H is the identity matrix we have a case of denoising, but when H is a blurring operator we are dealing with image deblurring. Image restoration is a typical ill-posed inverse problem and due to this ill-posedness the solution to the equation above is generally not unique. Different regularization methods like Tikhonov regularization and total variation regularization are often used to overcome this problem. The problem with these models is the oversmoothing of details in images. Sparsity-based models have emerged in recent years as a better option to find the best

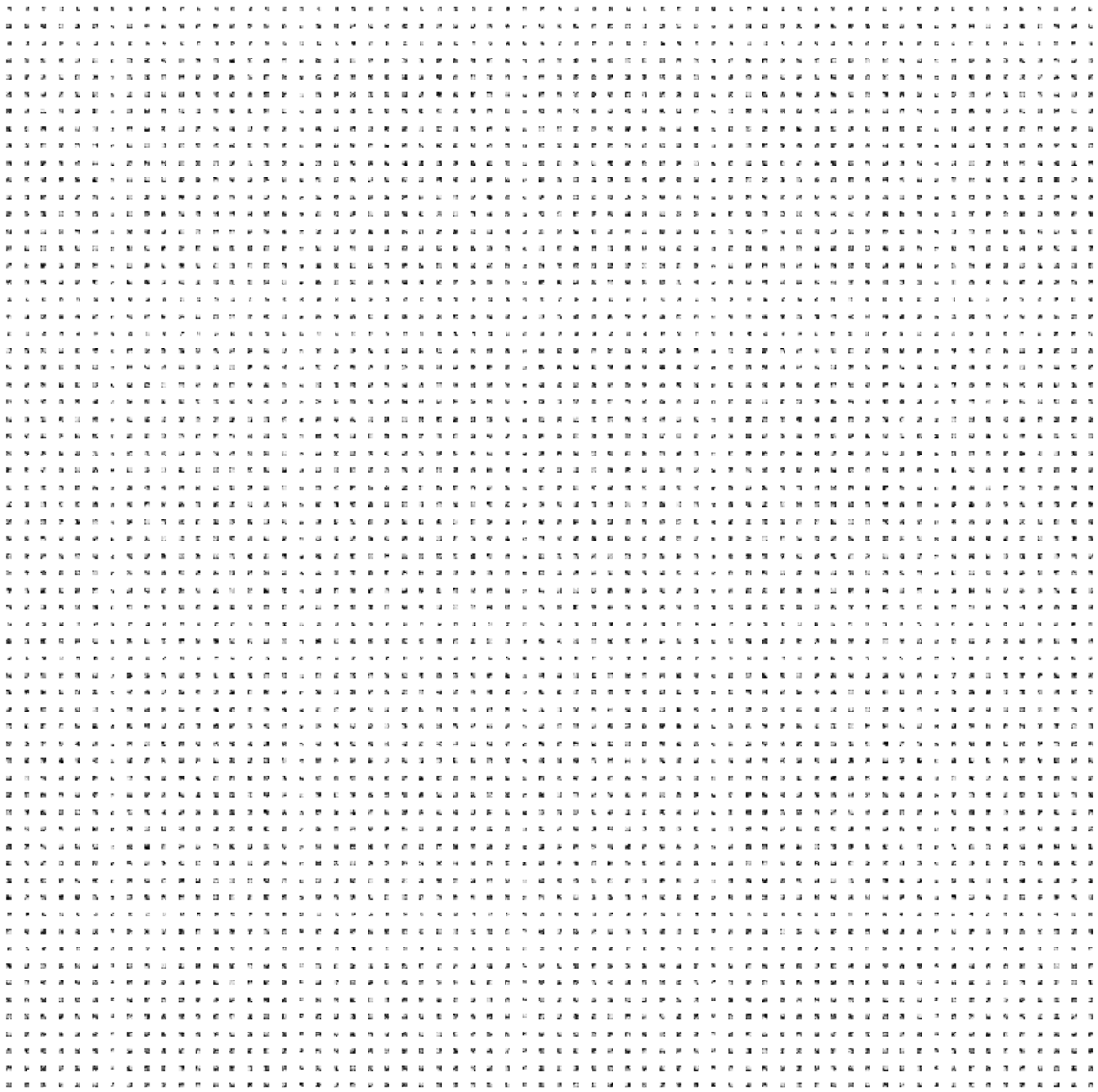

Figure 2.2: A 256×256 pixel image divided into 4×4 pixel patches.

possible solution to many image restoration problems.

Denoising

Let's have a closer look at one of the most common image restoration problems in image processing, denoising. We want to restore a clean image from a version that is corrupted with noise. The noise is often represented by additive white Gaussian noise (AWGN), as it is the type of noise that is often also naturally present in digital images due to internal or external aspects in the image processing pipeline [15]. Figure 2.3 shows an example of how the noise effects an image.

$$Y = X + noise \tag{2.2}$$

As before we will take a patch-based approach to this problem so the noise we added to the entire image will become noise in every single patch of the reference image X. All the patches will be in vectorized form again and they will form the columns of the matrices respectively.

$$y_i = x_i + noise_i \tag{2.3}$$

Typically AWGN has zero mean and a selected variance.

Figure 2.3: Illustrative example of the typical denoising problem.

2.3 Dictionary learning

In section 1.2 we saw that Dictionary learning offers the possibility to learn an adaptive dictionary D from a set of exemplars that we consider to represent the data in question well. One could also learn the dictionary straight from the data but we avoid this approach as it allows the possibility of committing an "inverse crime". The following discussion is based on [12, 7, 16].

Let the matrix $X \in \mathbb{R}^{m \times p}$ consist of a set of exemplar signals $x_i \in \mathbb{R}^m, i = 1, ..., p$ that are stored as the columns of the matrix. The goal of Dictionary Learning is to approximate the following matrix factorization problem:

$$\text{Find } D \text{ and } A \text{ such that } X \approx DA \tag{2.4}$$

where $D \in \mathbb{R}^{m \times n}$ is the dictionary made out of n prototype signals, called atoms, and $A \in \mathbb{R}^{n \times p}$ is the matrix whose i-th column is the synthesis coefficients vector a_i of the signal x_i in D, i.e. $X_i \approx Da_i$. The aim is to find D and A only from the given training data in X.

This is not a trivial task so to be able to even find an approximate solution to the problem we will need to use some kind of prior information. Most Dictionary learning approaches use as their main prior information the fact that A is sparse. As we saw in the introduction to this thesis, the redundancy in natural images allows us to assume there is a sparse way to represent these images. This form of approach is often referred to as sparse coding and due to this sparsity prior the dictionary learning task can be viewed as an optimization problem/sparse approximation problem:

$$\min_{D,a} \sum_{i=1}^{n} \|Da_i - x_i\|_2^2 \quad \text{subject to} \quad \forall i, \quad \|a_i\|_0^0 \leq L \tag{2.5}$$

where each example is a linear combination of atoms from D and has a sparse representation with no more than L atoms.

In our case we will be using the k-means algorithm as a form of vector quantization to find the dictionary that minimizes the error between the representation and the received signal matrix. When we have found a dictionary D each signal is represented by the closest atom (under Euclidian distance). We can write $x_i = Da_i$ where $a_i = e_j$ is a vector from the trivial basis, with all zero entries except a one in the j-th position. The index j is selected such that

$$\forall k \neq j \quad \|x_i - De_j\|_2^2 \leq \|x_i - De_k\|_2^2. \tag{2.6}$$

Because each example will be represented exactly by one atom of the dictionary, and the coefficient is forced to be 1, this can be referred to as a case of extreme sparsity.

The Mean-squared error (MSE) of the representation per each signal x_i is defined as $e_i^2 = \|x_i - Da_i\|_2^2$ and the overall MSE is

$$E = \sum_{i=1}^{p} e_i^2 = \|X - DA\|_2^2. \tag{2.7}$$

Now we can formulate the problem of finding a dictionary D that minimizes the error E subject to the limited structure of A, whose columns must be taken from the trivial basis as follows:

$$\min_{D,A}\{\|X - DA\|_2^2\} \quad \text{subject to} \quad \forall i, a_i = e_k \quad \text{for some } k. \tag{2.8}$$

The k-means algorithm is an iterative method which is used to train the dictionary for vector quantization. There are two steps for each iteration: one for evaluating A and another for updating the dictionary D. Let us next take a closer look at this algorithm that will play a major role in our restoration process.

2.4 K-means algorithm

The k-means algorithm is a commonly used clustering tool in data analysis. The algorithm divides the amount of observations n into k groups, that we call clusters, by minimizing the sum of the squares of the distances of each data point to its closest cluster centroid. This comes in handy when working with high dimensional data like our image patches because we cannot just plot the points and have a look at their spacial distribution.

Then again the Euclidian distance measure used by k-means may not be ideal for calculating distances in high dimensional spaces. Nevertheless we are specifically interested how well we can represent an image using this algorithm and use it for simple image processing tasks.

The k-means algorithm:

---

1. Choose the number of clusters k.
2. Choose the initial cluster centroids c_i, $i = 1, ..., k$ randomly.

3. Repeat the following two steps until convergence.

a) Assign each data point x to its closest cluster center,

$$\min_i \|c_i - x\|^2 \tag{2.9}$$

b) Compute the mean of all the data points in a cluster and replace the cluster center with the newly calculated mean.

Following these steps results in convergence but not definitely to an absolute minimum. The algorithm can also converge to a local minimum that is not optimal. This is why one should if possible run the algorithm several times with different random starting centroids and select the solution with the best result.

There are multiple ways to try and improve the basic k-means algorithm as suggested for example by [12]. We could change the initialization of the centroids to avoid empty clusters and to prevent a large number of centroids starting close together. In this thesis we will be using the default algorithm in MATLAB which actually uses the k-means ++ algorithm [17] for cluster initialization. We will also try the k-medoids algorithm witch is essentially the same algorithm but the representative of a cluster will be the closest actual data point to the cluster mean. So if in k-means the dictionary atoms are calculated means from the points in the cluster then in k-medoids the atoms are actual patches from the data.

2.5 SSIM vs. MSE

For a sensible comparison of two images we have to find a way to measure their similarity. As we don't have the time and resources to conduct a study that involves a large amount of actual people assessing the similarities of images we will have to rely on mathematics to help us compute a reasonable result. There are several ways to calculate the difference between two digital images and probably the most common and simple way is to calculate the Mean-squared error (MSE) of the images in question. Closely related to the MSE is the Peak signal-to-noise ratio (PSNR) which is also a commonly used measure in the

evaluation of the performance of image processing applications. The problem with these two measures is that they don't account for the human visual system. We see images and structures within these images differently than a computer might calculate the differences between pixel values. The Structural similarity index (SSIM) developed by Wang et. al [18] is designed to adapt better to the visual recognition of structural differences and the perception of the human eye and it will be our choice of measuring tool in the experiments to come. Still we have to keep in mind it is only another algorithm that has its own restrictions. And as stated before, to really be sure how the human visual system sees the difference between the images of interest, we would have to perform a large scale study on human test subjects. Let's have a closer look at the three different algorithms mentioned for an image f and a reference image f_0.

MSE

$$MSE = \frac{1}{nm} \sum_{0}^{n-1} \sum_{0}^{m-1} \|f_0(i,j) - f(i,j)\|^2 \tag{2.10}$$

PSNR

$$PSNR = 10 \log_{10} \frac{L^2}{MSE} \tag{2.11}$$

where L is the maximum possible pixel value of the image.

SSIM

The SSIM algorithm consists of three terms. A luminance, a contrast and a structural term. The exponents of the terms can be adjusted according to the relative importance of the three components. Multiplied together these terms give us the overall index value. The value will lie between -1 and 1, 1 being the highest value and only achieved if the two compared images are identical.

$$SSIM(f, f_0) = [l(f, f_0)]^\alpha \cdot [c(f, f_0)]^\beta \cdot [s(f, f_0)]^\gamma \tag{2.12}$$

where

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1},$$
$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2},$$
$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

where μ_x and μ_y are the means of the image signals, σ_x and σ_y are the standard deviations (square roots of variance) of the signals and σ_{xy} is the cross-covariance of the images. The constants $C_1 = (K_1L)^2$ and $C_2 = (K_2L)^2$ are added to avoid instability. L is the dynamic range of the pixel values and the small constants $K_1 \ll 1$ and $K_2 \ll 1$ assume default values of 0.01 and 0.03 respectively. Using also default selections for the exponents in (2.6) $\alpha = \beta = \gamma = 1$ and for the third constant $C_3 = \frac{C_2}{2}$, the Structural Similarity Index simplifies to:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{2.13}$$

We will use the SSIM to compare the original images to the different reconstructions in this thesis.

2.6 Pre-processing

When handling data, it is usually foremost to pre-process the data according to the requirements and principles of algorithms used later on. Unfortunately, the literature is not clear about which pre-processing step should be applied given a specific problem at hand [2]. In our case, as we will be using the k-means algorithm to cluster image patches, we will use two simple pre-processing steps that are generally called centering and normalizing.

First we center the data by simply subtracting the means of each image patch individually. The overall effect of this operation is to amplify the high-frequency structures such as edges, resulting in more high-frequency patterns in the learned dictionary [5]. We can see the effect of centering each patch individually in Figure 2.4. We see clearly how the higher frequencies are more prominent in the centered image.

Figure 2.4: Image centered (mean subtracted) each patch individually and original image.

Then we normalize the patches to have values between -1 and 1 by dividing all patches with the absolute maximum value of the patch data. We normalize all used patches to be sure our values lie in the same range and so we can calculate with them correctly.

Chapter 3

Experiments

In this chapter we will go through the actual experiments, take a look at the results and discuss them. All of the experiments are run in MATLAB (R2016a) 64-bit Windows version on a PC workstation with a Intel CPU 3.40 Ghz processor and 8Gb of memory.

Let's first have a look at the data we will run our experiments on. We will use two different sets of images. As the first set of data we use the peppercorn picture dataset provided by the Industrial Mathematics Laboratory of the Department of Mathematics and Statistics of University of Helsinki (http://fips.fi/photographic_dataset.php). It consists of 10 high resolution images taken at the Industrial Mathematics Laboratory by Teemu Helenius and Prof. Samuli Siltanen. A detailed report of the collected photographic data in question can be found in [19]. The images are of peppercorns that have been randomly spread on a horizontal platform and photographed from straight above.

As a second data set we use pictures of buildings taken by the author outside, on the streets of Helsinki on October 19, 2016. The images were shot with a Canon 5D Mark II camera and a Canon EF 17-40mm f/4L USM lens. The green color channel was selected in Corel PaintShop Pro X8 (as it gave the best looking result) to represent the images in black and white so we were left with only one value for every pixel.

We choose 8 peppercorn images (Figure 3.1) and 12 building images (Figure 3.2) to compile a large data set of image patches for each of the images sets. We leave some of the images out of the learning phase to use them as reference images later on and avoid committing an inverse crime.

Because especially the original peppercorn images are really large we have to downsample them so that the ratio of the patch size to the size of the peppers is reasonable.

This means that we can be sure a single patch can contain valuable visual/structural information of the image. We also crop the images to be square and choose the highest power of two that fits a single image to avoid having to pad the images and enabling multiple downsampling steps next. So we crop a 4096×4096 sized part of the chosen images. In the original peppercorn images one peppercorn fits approximately into a 64×64 patch. Downsampling three times by 2×2 we have a pretty good fit of peppers in patches and an image size of 512×512 pixels. The 8 images seen in Figure 3.1 are actually the downsampled ones. We proceed in similar fashion with the second data set so that we are again left with 512×512 sized images.

Then we divide these images into patches of the chosen size just like in section 2.1. After dividing the 8 downsampled peppercorn images into 8×8 patches we are left with a matrix $X \in \mathbb{R}^{64 \times 32768}$, that holds the individual patches as columns. Similarly we get a matrix that holds the building images patches that is of size 64×49152.

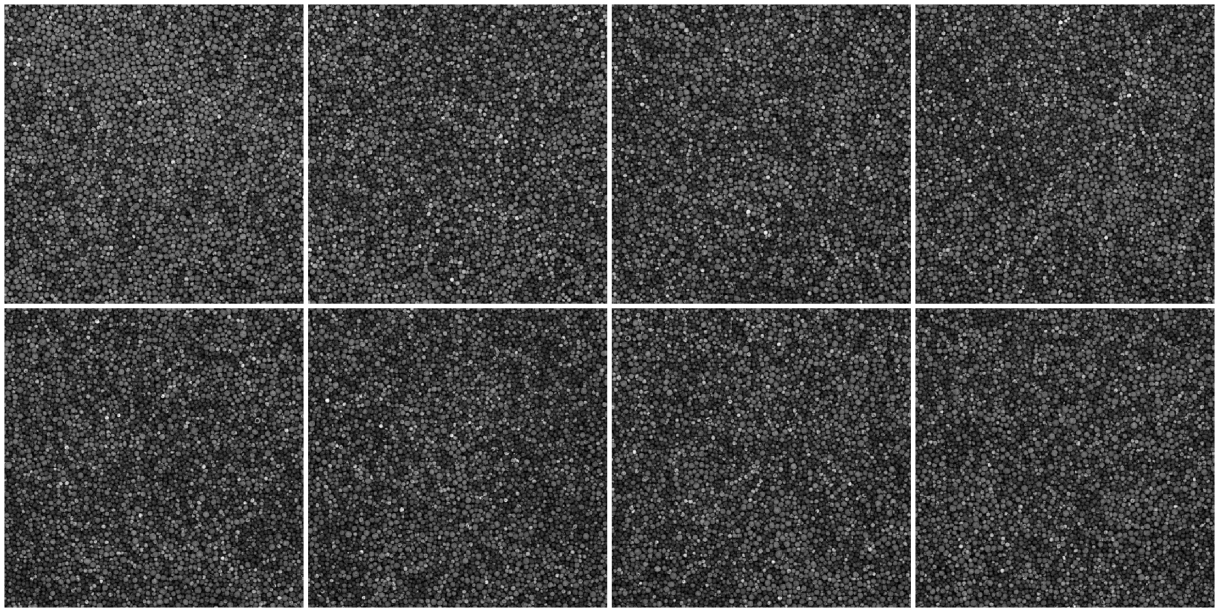

Figure 3.1: The 8 downsampled peppercorn images used to learn the dictionary.

Pre-processing

The next step is to pre-process the patch data. The pre-processing is done like in section 2.6. First we center the data i.e. subtract the means of each patch individually and then we normalize the data.

Figure 3.2: Dataset consisting of images of various buildings in Helsinki.

Dimensionality and PCA

Before we jump into the learning of the dictionaries let's have a look at our patch data. The original images of 512×512 pixels can be thought of as points in 262144-dimensional space. Likewise the 8×8 patches for example can be presented as points in 64-dimensional space. Since we know that the data in images is redundant we want to try to find out what the intrinsic dimension of the patch data is i.e. if we could represent the data without all 64 dimensions. With the help of Principal Component Analysis (PCA) we can count the amount of significant eigenvalues and estimate the dimensionality of the data. This method works as long as our data is linear and noiseless.

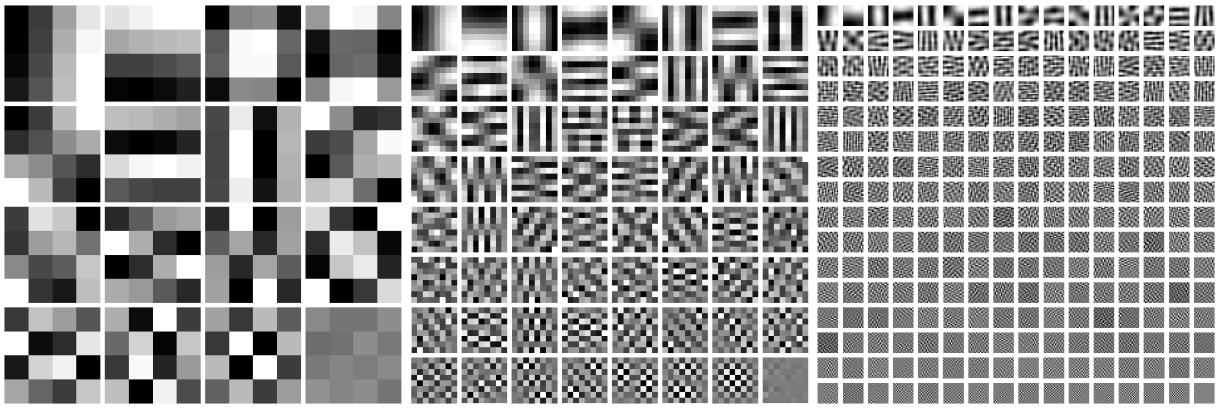

Figure 3.3: Principal Components of 4x4 (left), 8x8 (middle) and 16x16 (right) image patches from the building image data.

If we have a look at the actual principal components in Figure 3.3 we notice that the last components don't really seem to hold a lot of meaningful structural information. This is due to the fact that the variances of the smaller components are too similar [6].

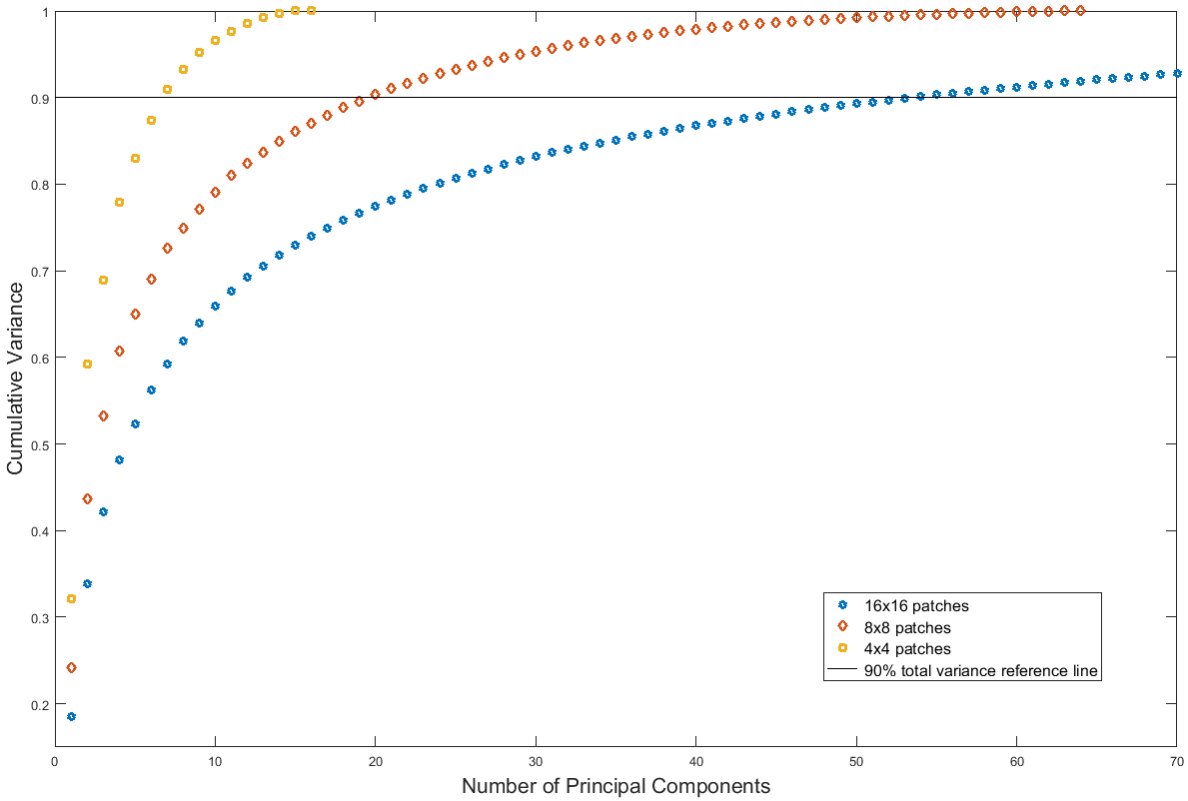

Figure 3.4: The cumulative variance of the first principal components for different sized image patch data.

Figure 3.4 shows us the cumulative variance of the principal components of the image patch data for the three different patch sizes. We see that the first 7 principal components of the 4×4 patches account for over 90% of the variance. For the 8×8 patches the first 20 components out of 64 account for 90% of the variance and for the 16×16 patches the amount of components needed to reach this value is 54 out of the total 256. This tells us that there is a considerable amount of redundancy in the data and the larger our patches get the smaller percentage of total components we need to account for a certain amount of variance in the data.

The authors of [20] state that patches sampled from natural images are highly structured and constitute a low-dimensional subset of the high dimensional ambient space and in fact, natural image patches are commonly assumed to lie close to a low-dimensional manifold. This means that the Euclidian distance we use to measure distances between our data points might not be the best option. In the future it might be interesting to explore other distance measures as well but we won't include these considerations in this thesis. If one is interested in the manifold structure of image patches and manifold models

look for example in [20, 21].

So with help of PCA we saw that the patch data is structured in a fashion that most of the variance of the data can be represented in a fraction of the total dimensions of the data. Usually this would be helpful for dimensionality reduction for example in image compression tasks. We on the other hand want to keeps this result in mind as we will be looking to find at a good value for the amount of clusters for clustering our data.

3.1 Learning a dictionary (by Applying k-means on the patch data)

The next step is to learn a dictionary from our patch data. We want to cluster our data into groups of similar patches. We will be using the k-means clustering algorithm described in section 2.4. But how should we choose the number of clusters k ? How many different groups are there? There is no clear answer to this and a good choice for k often depends on the data at hand. In section 1.2 we stated that the dictionary is often chosen to be overcomplete, even though fever dictionary atoms than patch dimensions can lead to reasonable results.

A common value for k in many cluster based dictionary learning methods is 256. We can have a look at Figure 3.5 that presents how the total sum of within cluster distances changes as the number of clusters k increases. The graph looks steadily decreasing and logarithmic and there is no so called elbow point to be found to help us make a decision on how to choose k based on this information.

Next let's have a look at some different sized dictionaries learned from our two datasets. The selected values for k are 4, 9, 16, 25, 36, 64, 121 and 1024. Figure 3.6 shows the dictionaries learned from the peppercorn images and Figure 3.7 holds the dictionaries learned from the building image data. The atoms in the individual dictionaries in these figures are sorted from left to right and top to bottom in descending order by the total number of patches that belong to that particular cluster. This means the centroid from the cluster with the most patches in it will be in the top left corner and so on.

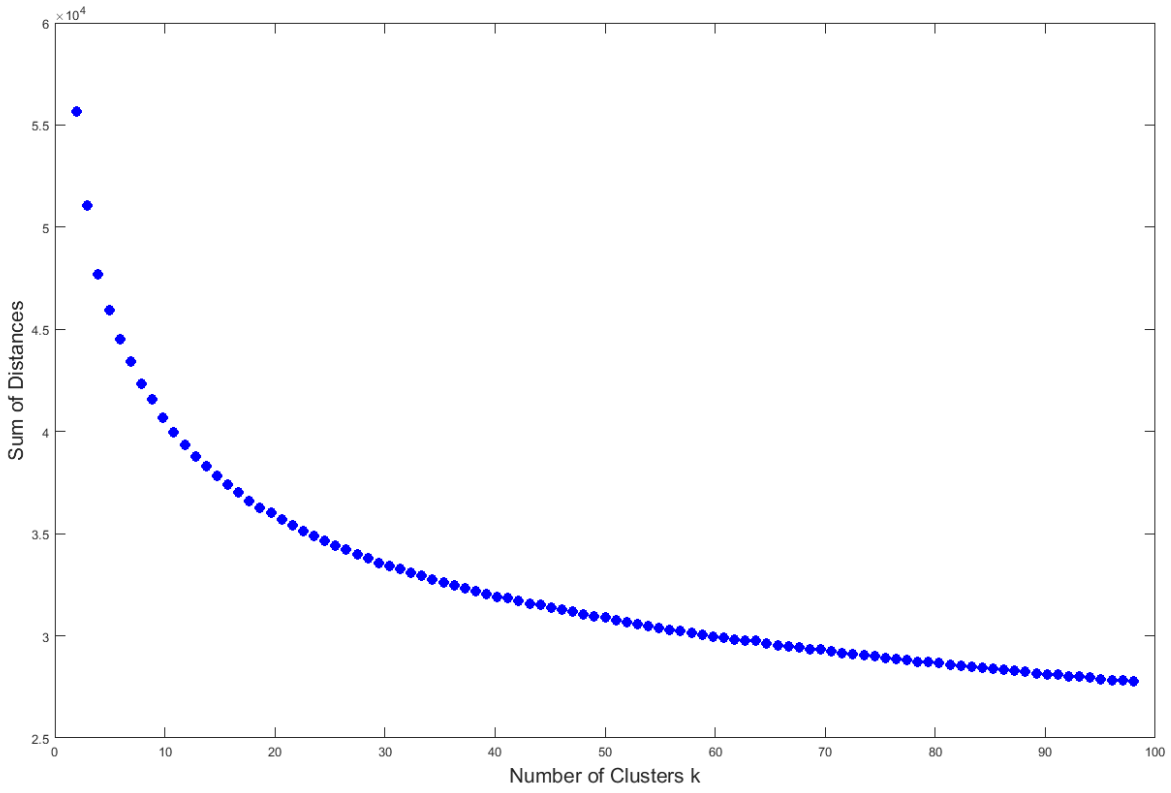

Figure 3.5: The total sum of distances as a function of the number of clusters k.

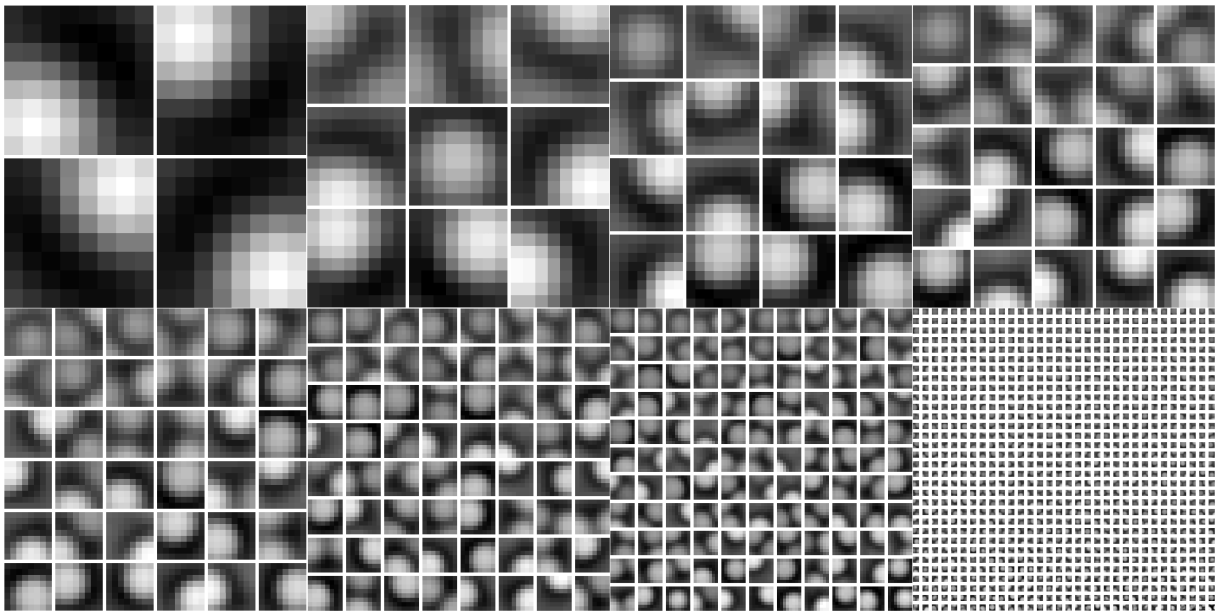

Figure 3.6: Different size dictionaries learned with k-means from 8×8 image patches of the peppercorn images.

Looking at the peppercorn dictionaries, beginning from the smallest, we notice that the atoms try to cover all possible options for the location of a peppercorn in a single

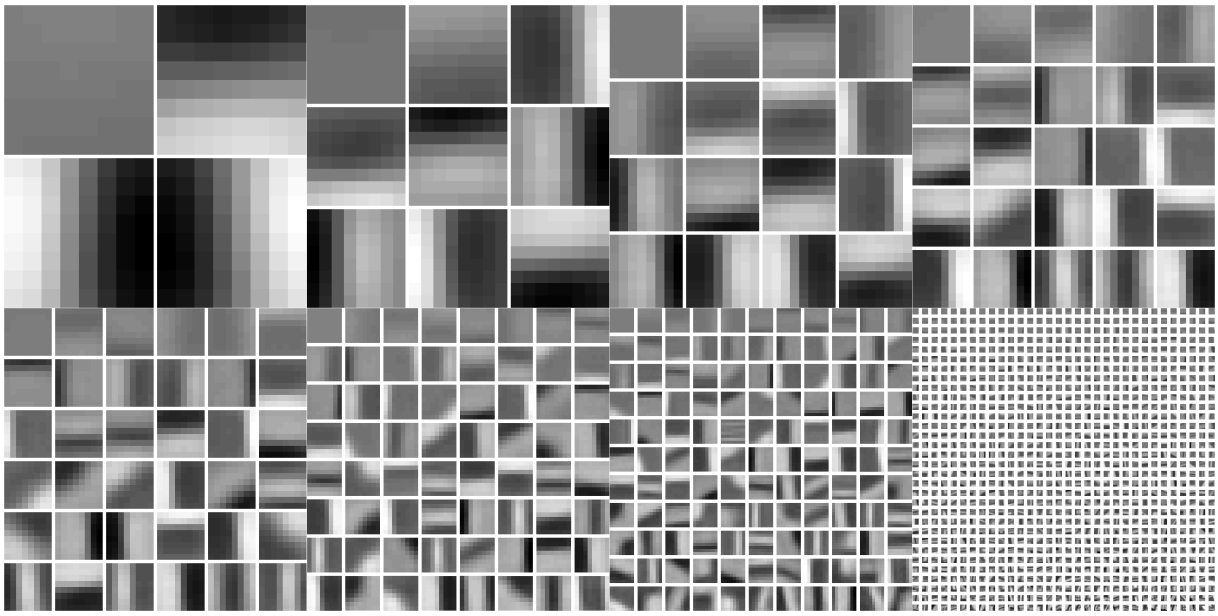

Figure 3.7: Different size dictionaries learned with k-means from 8×8 image patches of the building images.

patch. In the case of four atoms the peppercorns are located in each of the four corners of the patches. The larger our dictionary grows the more varied the atoms get. Not only considering the location of the peppercorns but also the different shades of gray of the atoms and the size of the peppercorns. Seems obvious that with a larger dictionary, containing a more varied selection of atoms, we will be able to reconstruct an image better.

In the building data dictionaries the most dominant atom is an almost uniform one. Not surprising, since the data consists of a lot of uniform parts, like the sky and walls. In addition to this uniform atom we see vertical and horizontal edge like atoms. As the atoms grow more varied in the larger dictionaries we see some diagonal edges as well and in the two largest dictionaries we notice even small structures and patterns.

3.2 Reconstructing an image

Now we can use these dictionaries to reconstruct images from our data sets that we did not include in the dictionary learning process. We take for example an unused image of the peppercorn set and downsample it to be the same size as the original patch data images (512×512 pixels) and divide it into same sized patches as well. We then center and normalize the patches as we did with the patches for learning the dictionary before. We store the subtracted mean values for each of the patches, as we will add them to the

resulting image later on in the process. Then we calculate the relative error (Euclidian distance) between the new patches from the image we wish to reconstruct and the cluster centroids (dictionary atoms) that we found using the k-means algorithm and replace each patch of the reference image with the best match from our dictionary. Then we add the means of the reference patches back to the newly found replacements from the dictionary.

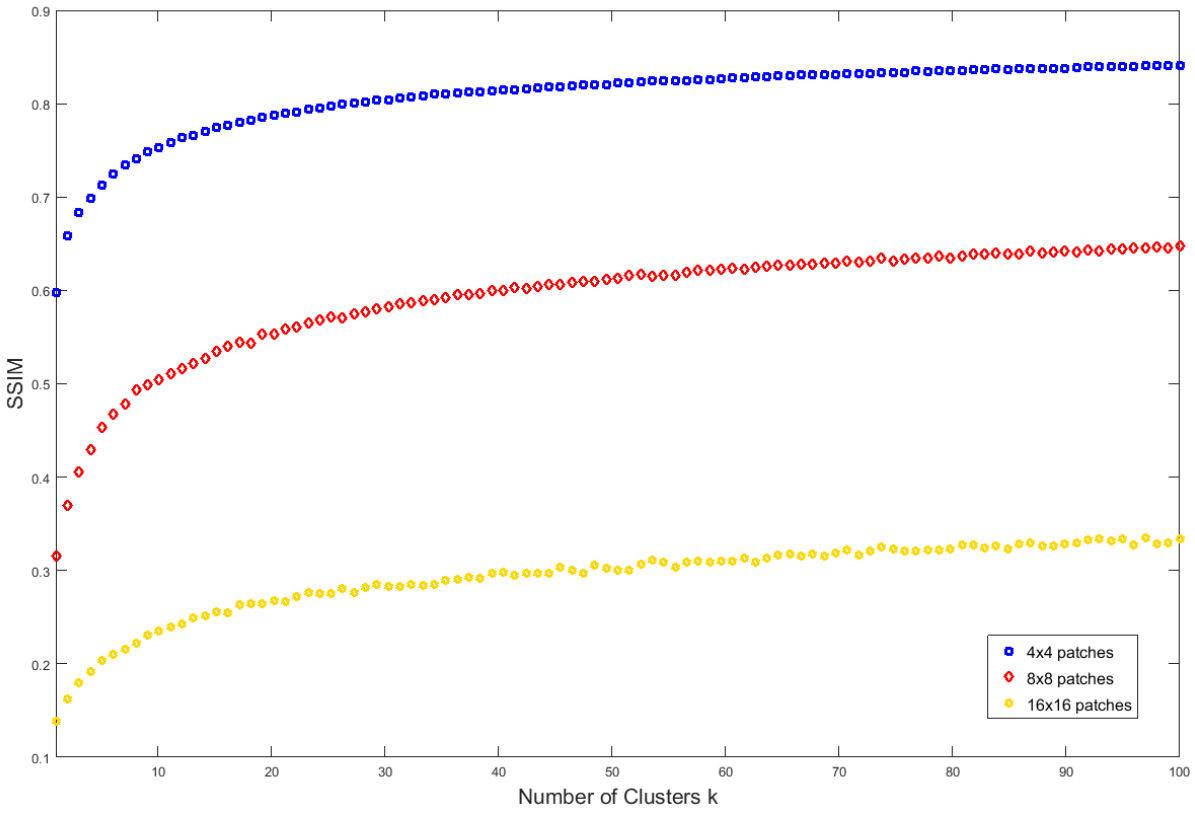

Figure 3.8: SSIM values between the original and the reconstructed image as a function of the amount of clusters k with patch sizes 4×4, 8×8 and 16×16.

We can now calculate the SSIM between the original image and the reconstructed image and see how it changes as a function of k, the amount of clusters used. The results can be seen in Figure 3.8. We see that the SSIM values improve significantly in the beginning but as the number of clusters grows the improvement slows down. The SSIM values grow gradually larger but the amount of clusters has to increase a lot to see any significant improvements. We do not want to choose the ideal amount of clusters only by looking at the plotted SSIM values. Let us look at the actual reconstructions with 8×8 patches and see how they compare to the original image. The animation 3.9 shows us the reconstructed images, with increasing number of atoms in the dictionary used, on the left and on the right the original reference image. Certain values for k were selected by the

author to portrait the fast improvement of the reconstruction in the beginning and the need for a really large k value to see big improvements later on (without the animation getting too large to be included in this pdf). We see that in order to get a really good approximation the amount of dictionary atoms needs to be quite high but also that after a while the benefit of increasing the amount of clusters doesn't pay off as a significant improvement of the reconstruction anymore. We could increase the amount of clusters to be really high but the computational cost increases as well. At some point we would need an even larger amount of exemplary patches to start with to be able to further increase the number of clusters.

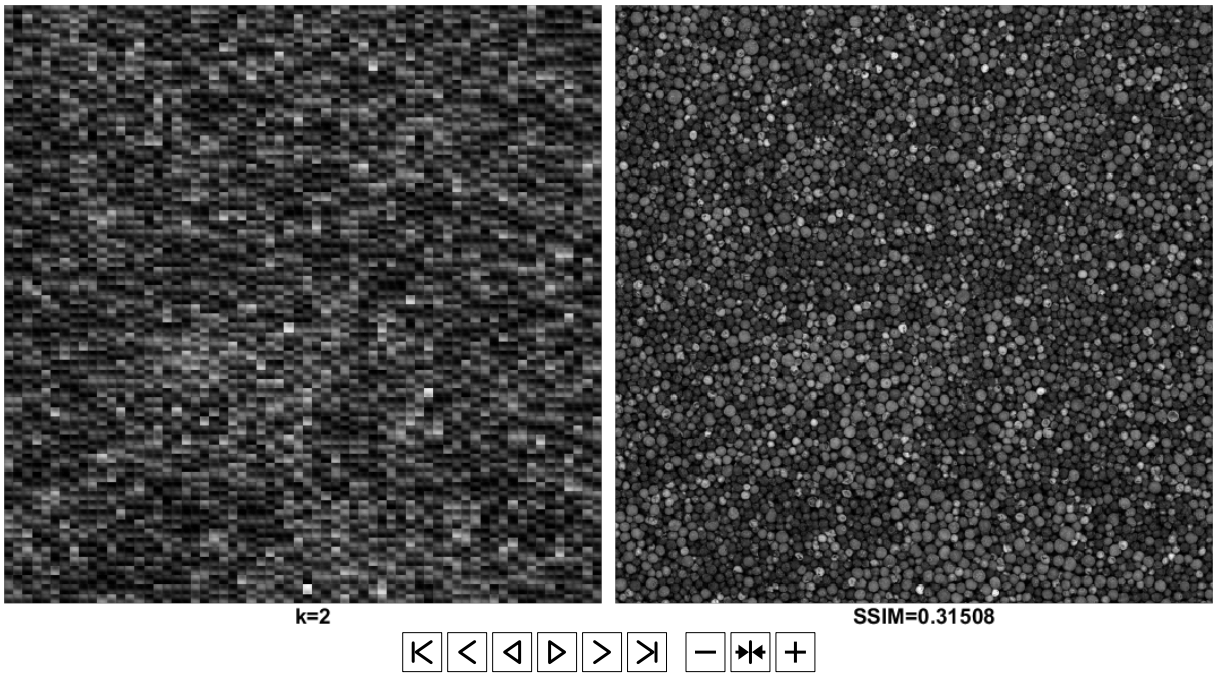

Figure 3.9: Animation showing the improvement in reconstruction of an image with different sized dictionaries with 8×8 atoms .

It would seem reasonable to choose the number of clusters to be large enugh to get a satisfying reconstruction result but not too large because the benefits won't overpower the cost of possibly needing a larger patch data and the longer computing time of learning a huge dictionary.

We saw that in the case of a clear and almost optimal reference image we are able to find good fitting atoms to replace patches of the image. In fact from Figure 3.8 we see that the smaller the patches the better matches we find as the resulting image is closer to the original. In the trivial case we could replace each individual 1×1 pixel value with the closest match and end up with an almost identical image. As most image processing

applications don't work with the clean original image but with a noisy or blurry version of it we will proceed to this problem next.

3.3 Image Denoising

We saw how well we are able to reconstruct an original image with the help of a dictionary built using the k-means algorithm. What if the image we try to reconstruct is noisy? How does the same method work for denoising purposes?

The peppercorn and building image data we use do not have visible noise per se because of the low ISO rates and relatively short shutter speeds used. To simulate the situation where our image signal to be restored has some noise in it we will have to add it to the image. The noise in digital images is often simulated by additive white Gaussian noise. Our range of values for the images is still between -1 and 1 as we have scaled the values so before. We add noise with standard deviation σ of 0.1 to the peppercorn reference image first. Figure 3.10 shows the resulting noisy peppercorn image in comparison with the original image.

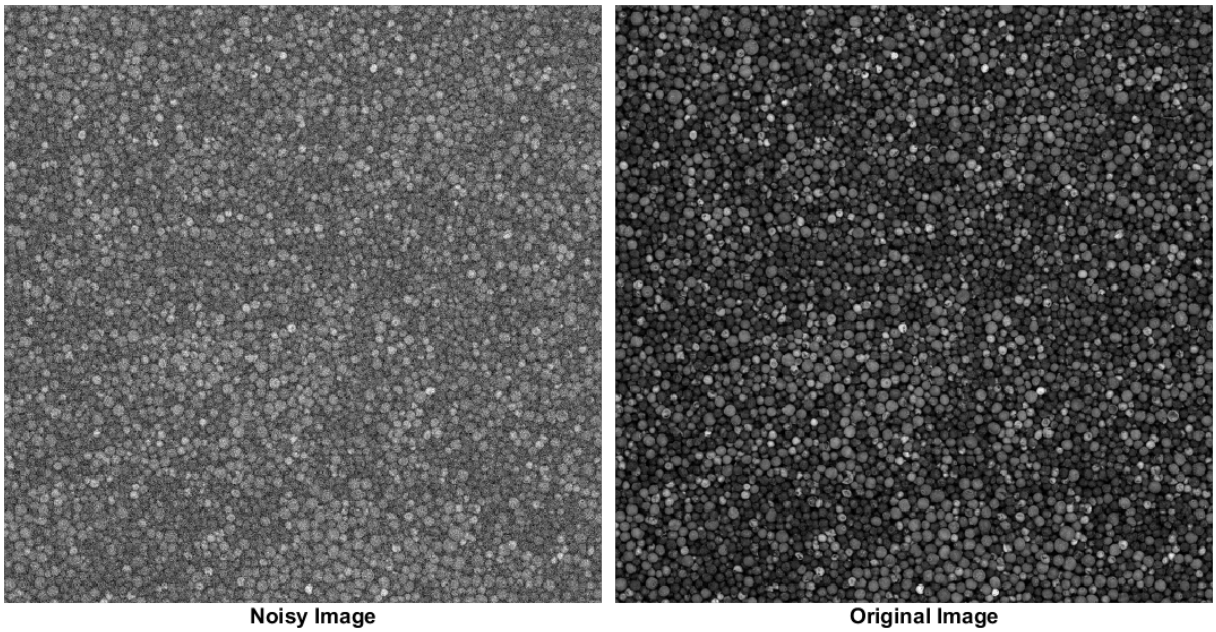

Figure 3.10: Comparison between a noisy image of peppecorns and the original.

So let's try to reconstruct the image by using the same dictionaries as before, learned from noise free exemplars. Figure 3.11 shows the denoising results for the peppercorn image. We see how the SSIM index is clearly smaller for the noisy reconstructions with

the smallest 4×4 patches but the larger our patches are the better the reconstructions is compared to the unnoisy ones. With 16x16 patches the SSIM values are actually almost as good. This would suggest that for image denoising applications it is preferable to use larger patches if possible because the reconstruction does not suffer additional loss in structural similarity.

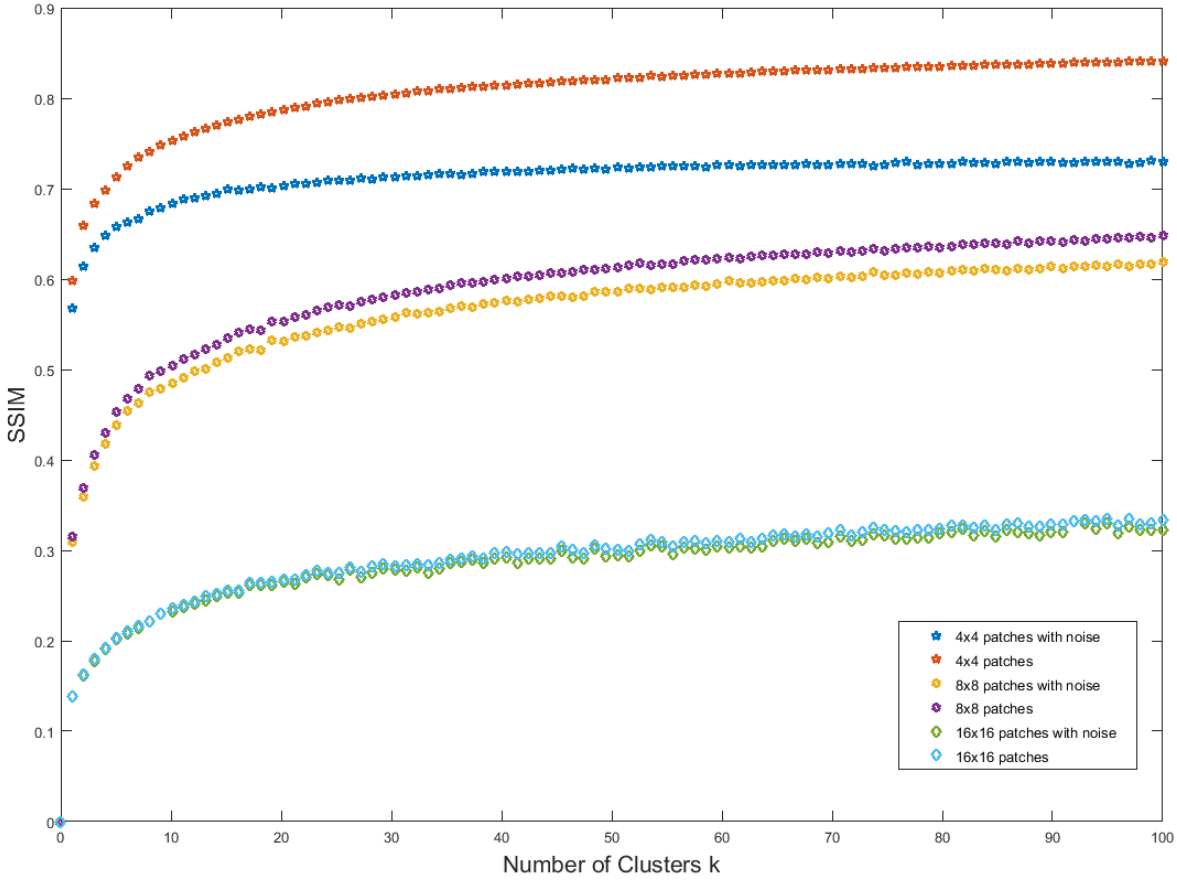

Figure 3.11: SSIM values between the original and the reconstructed peppercorn image as well as between the original and the denoised image as a function of the amount of clusters k. Patch sizes 4×4, 8×8 and 16×16.

For the building image dataset we did the same experiment but with two different values for the standard deviation of the noise ($\sigma_1 = 0.1$ and σ_2 of 0.05). Figure 3.12 shows the results of these computations. The results are similar compared to the denoising of the peppercorn image, as they should.

Another observation is the fact that the overall SSIM values are higher compared to the peppercorn reconstructions, especially with the larger patches. This difference can be explained by the fact that the building images hold structures of different scales and the most prominent ones fit in the larger patches as well. The peppercorns were simply

too small for the 16x16 patches. We also see how the smallest 4x4 patches will fail to represent the actual structures of the image, especially with higher noise levels. The residual noise in the reconstructions is visible even with the 8x8 patches (Figure 3.13). The small patches are not robust enough to deal with the noise.

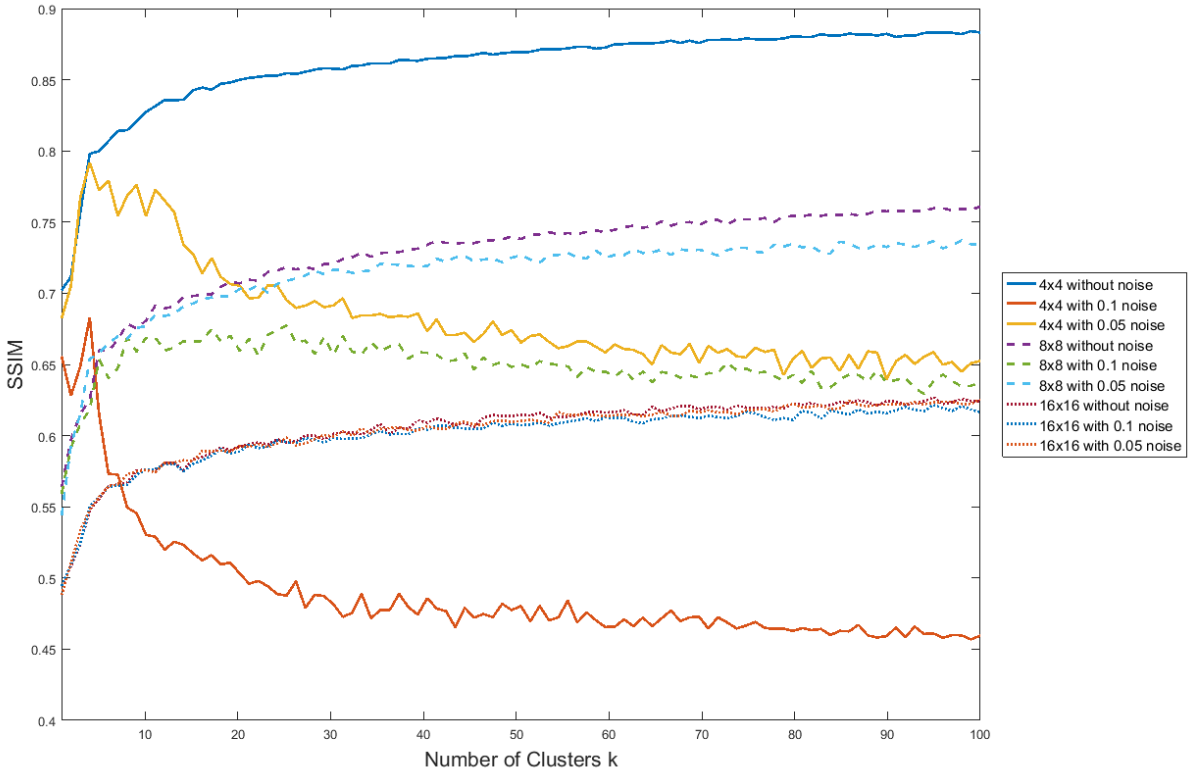

Figure 3.12: SSIM values between the original and the reconstructed house image as well as between the original and the denoised images as a function of the amount of clusters k. Patch sizes 4×4, 8×8 and 16×16.

As the patch size increases we also have less patches from which to build the dictionaries as the amount of exemplary images is constant in our case. But it seems like we have enough data to represent the images well enough. The more patches and dictionary atoms the better the resulting reconstruction seems to get if the patch size is large enough and the amount of noise in the data is not too extreme.

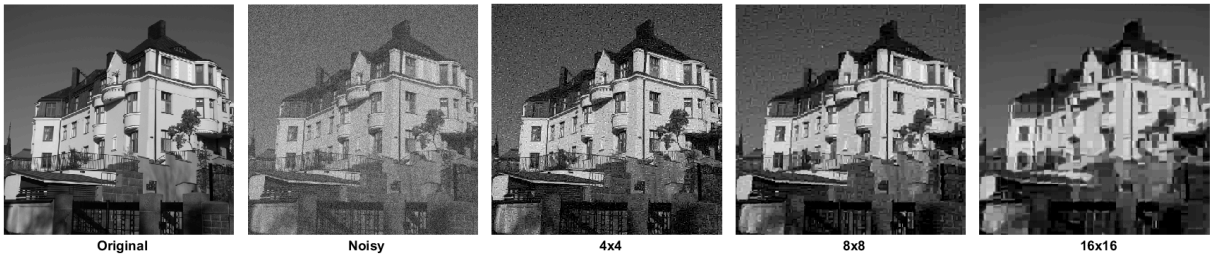

Figure 3.13: Comparison between the original and the noisy image of the house and reconstructions with different patch sizes with k=64 clusters used to build the dictionary and $\sigma_1 = 0.1$.

3.4 Other experiments

In this section we present results from additional experiments conducted during the making of this thesis.

Constructing an image out of a dictionary learned from different examples

Previously we used dictionaries learned from images that are similar to the image we tried to reconstruct. What if we learn a dictionary from totally different natural images? Let's say we reconstruct a building image by using the dictionary learned from our pepper pictures and vice versa.

The resulting images of these reconstructions are quite fascinating. Figure 3.14 depicts the house made out of peppercorns and Figure 3.15 the peppercorns made out of houses. We can clearly see that the atoms in the reconstructed house have the round shapes of the peppercorns. Still if the patch size selected is small enough, compared to the prominent structures that distinguish the main motive of the image, it is clear that we will be able to reconstruct these structures well enough to recognize what the image represents.

Figure 3.14: "House of Peppercorns"

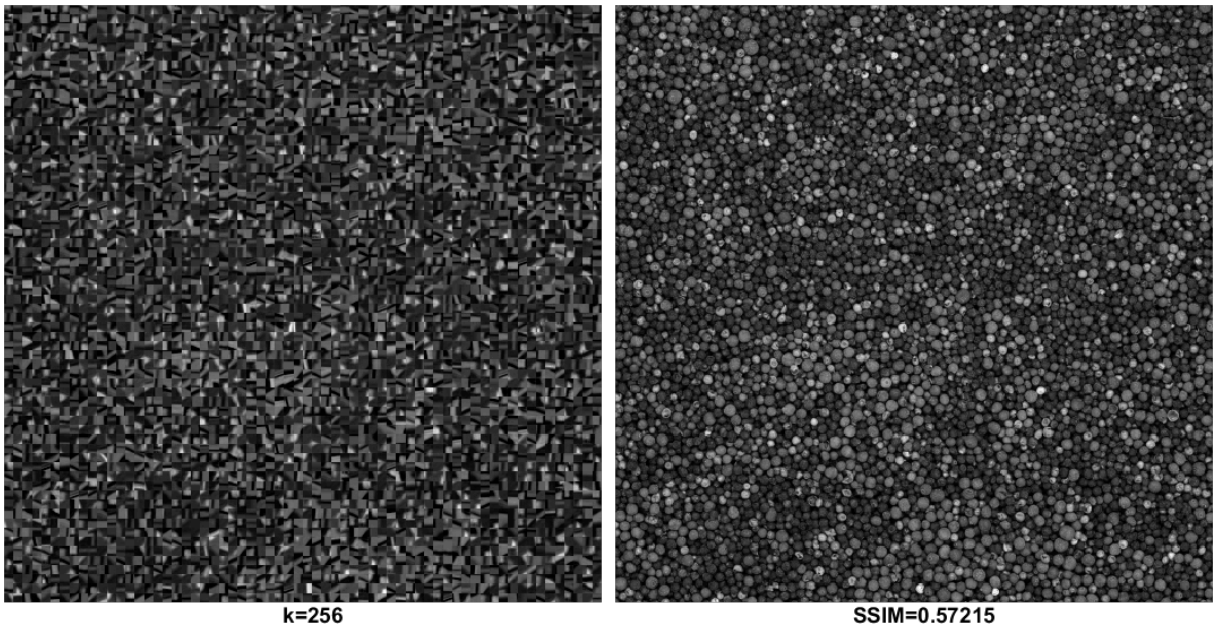

Figure 3.15: "Peppercorns of Houses"

Then again if the structures are small like the peppercorns we run into difficulties to exactly restore these finer structures. We see that the darker and lighter areas are well distinguishable in Figure 3.15, but if we zoom into the image we see that you could not tell if the image represents peppercorns, a piece of modern art or just random patches that don't seem to fit together. We can obviously see that the building image reconstructed out of peppercorn patches is a house. But if we would look at just the fine details of the

house image, for example just one window, we would again probably not be able to tell what it is.

It is interesting to see that the SSIM value is actually better with the peppercorns made out of buildings than the house of peppercorns, even though you can clearly make out the house in the image but not really the peppercorns in the other one. This is most likely because the SSIM index looks only at smaller structures in the image (the algorithm usually works with 8x8 or 11x11 sized windows). We see that the prominent edges of the house image are large and even though on a small scale the similarity is not so good the structure of the house can be represented well with almost any dictionary. This is also due to the fact that the means that are subtracted of the individual patches of the reference image are added back to the resulting images. We can actually see from Figure 3.16 that the mean values already hold some structural information of the images. We can detect the house in the left image and we see that the SSIM value actually decreases when using the peppercorn dictionary to the value of just the means. With the peppercorn image the SSIM value of the mean image is very low and the building image dictionary helps to increase the similarity quite a lot even if it is not able to reconstruct small individual peppercorns.

Still we must conclude that the learning of a dictionary out of patches from similar images to the original gives a much better result. The knowledge of the mean values helps us to come up with some sort of reconstruction for any dictionary by providing the distinction between lighter and darker areas but to represent the details and fine structures we need a more specifically learned dictionary.

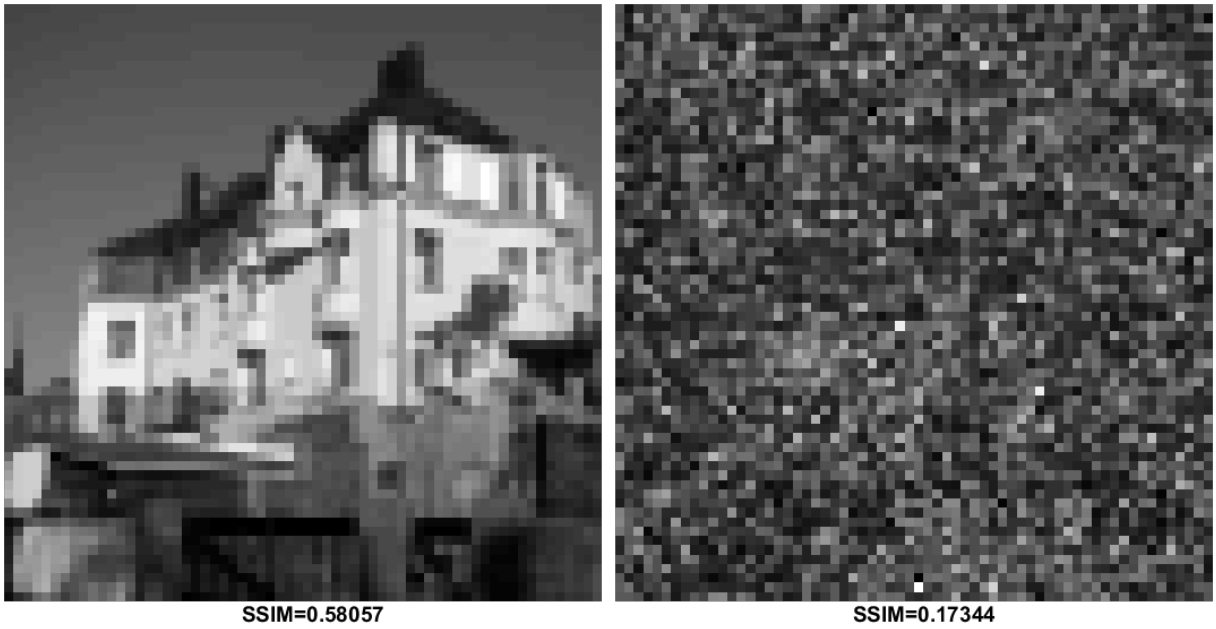

Figure 3.16: Only the mean values for each individual patch of the two reference images and the SSIM value compared to the original images.

SSIM index maps

As the last part of this section we want to take a look at some SSIM index maps. The SSIM maps reveal areas of the restored images with the biggest structural differences compared to the original image. The lighter an area appears in the SSIM map the better the similarity and the darker a region appears the larger the dissimilarities.

Figure 3.17 depicts the structural similarity maps between an original image, a noisy image and the different denoising results for reconstructions with different sized patches identical to the situation in Figure 3.13 we saw before. We also have the according SSIM values for comparison. The original images SSIM value compared to itself is of course 1 and the according SSIM map is completely white. The SSIM map of the image with added noise shows us that the outlines of the main structures are the most similar to the original but the more constant areas suffer quite a lot from the noise. As we move from the 4x4 reconstruction to the 8x8 and finally to the 16x16 reconstruction we see how the SSIM map evolves almost into a negative as the sharp edges become more feebly portrayed but at the same time the effects of the noise are overcome in the constant areas.

The SSIM maps from our "House of peppercorns" and the "Peppercorns of houses" images from the previous section are shown in Figure 3.18. These maps confirm the fact

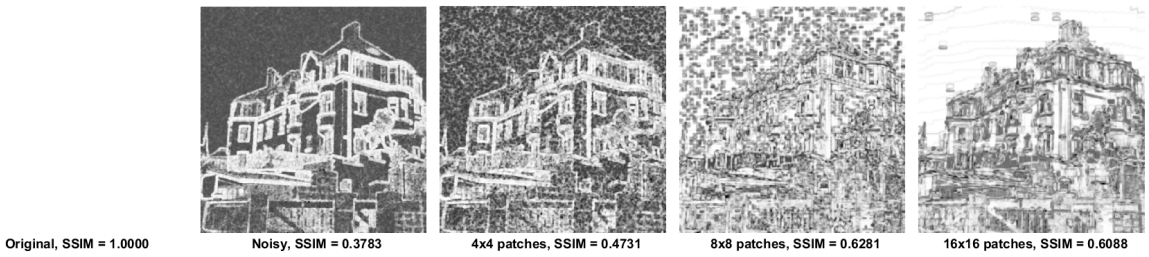

Figure 3.17: SSIM maps of the original and the noisy image of the house and reconstructions with different patch sizes with k=64 clusters used to build the dictionary.

that we see the outlines of the house clearly but the small round shapes of the peppers are hard to recover with a dictionary from the house image patches.

Then again we notice how the sky and other constant regions of the house image are not well represented with round pepper shapes according to the SSIM index as these areas are rather dark. So by looking at these SSIM maps it does not surprise us that the overall SSIM value was better for the "Peppercorns of houses" reconstruction. We can also conclude that it would be fairly easy to improve the similarity of the "House of peppercorns" by introducing just one constant atom to the dictionary used. To improve the "Peppercorns of houses" image we would need several new atoms that include the information of suitably sized round shapes.

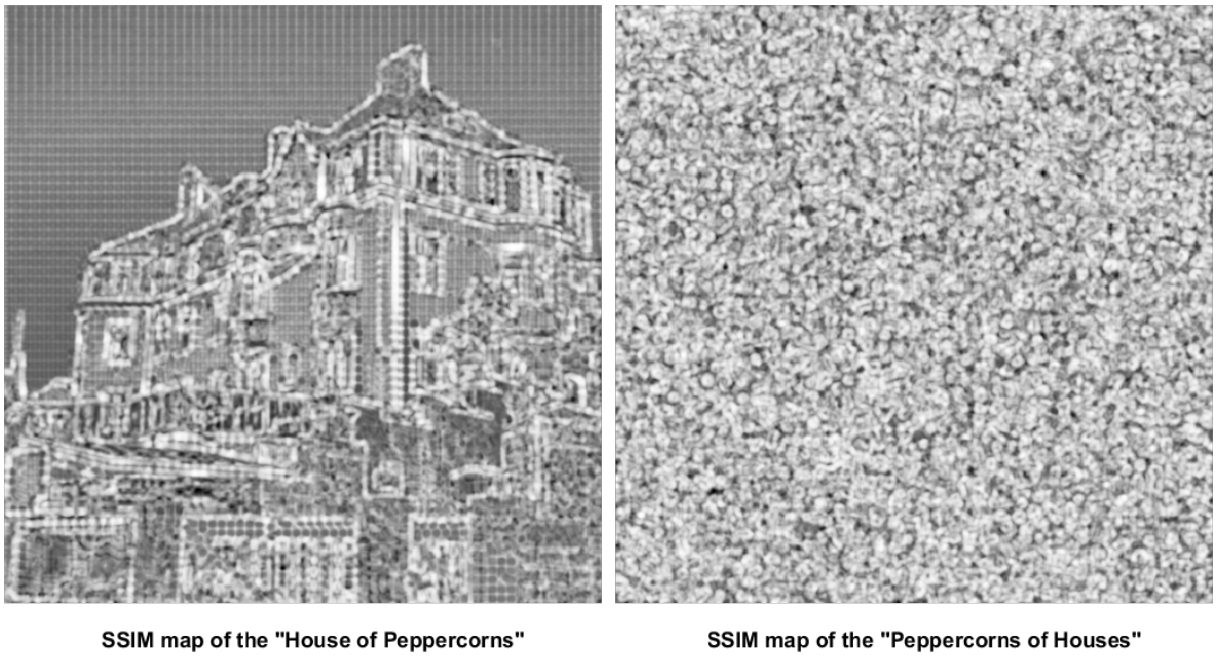

Figure 3.18: "SSIM maps of the "House of peppercorns" and the "Peppercorns of houses".

Chapter 4

Conclusion

During the course of this thesis we have seen how to use a dictionary learned from image patches to reconstruct and denoise images. We did this in a very simplistic manner by using the k-means clustering algorithm. Our direct and simple approach leaves clearly room for improvements but also reveals the basic principles of the patch-based dictionary learning methods.

Because the dictionary atoms we used are square patches that have to go in exact predefined places and are not allowed to overlap, we see prominent blocking artifacts in the restored images. It is hard enough to find a matching atom but to find two that fit in place and fit together side by side is even harder with the simple method we use. This can be seen also in parts of the images that have relatively thin, straight lines that are not exactly horizontal or vertical. It would be interesting to look into ways to improve these regions between patches in the future.

The simplest way to reduce the patch artifacts would be to post-process the images by applying a suitable filter but this would most likely lead to blurring and oversmoothing of edges and highly textured structures. Perhaps a better solution would be trying some sort of image inpainting method to deal with the border areas between patches only.

It also seems difficult to restore small details, details clearly smaller than the used patch size, with a reasonably sized dictionary. Especially texture details that lie exactly on the border between two patches in the original image are problematic to restore. We learned that after a while the structural similarity between our reconstructions and the original images doesn't improve that significantly anymore when increasing the amount of atoms in the dictionary used. This is probably due to the fact that the structural differences

left are mostly small details that the dictionary atoms can't describe anyway because of their predefined size and the inflexibility described earlier that leads to problems between patches.

As a next step in developing a method for image restoration we might want to look into better ways to construct the image out of the dictionary atoms. Instead of simply placing the atoms in predefined places we might want to consider placing them in a more flexible way. Also the way the patches relate to their neighbors is an area of improvement. Perhaps we could take the best suiting atom first and start constructing the image from that area but not limit the method to predefined square areas but use perhaps Voronoi diagrams or other more flexible methods to tile the entire image. One could even give weighted values for the patches based on the goodness of fit.

Another way to go would be enabling the patch sizes to vary in our construction so that areas with smaller detailed structures would be reconstructed by using smaller patches and larger structures with larger patches. But we have to remember that the small patches do not work with really noisy images.

We used the prior knowledge of the type of image we had to learn a dictionary from similar images. This *a priori* knowledge is usually available in image processing so it is legit and wise to confine a method to a certain set of images. Still it would be nice to come up with a more universal approach that works well with all kinds of data but since image spaces tend to be quite high dimensional and natural images seldom occupy these spaces uniformly it is only natural to find ways to confine each individual image processing problem to a more manageable surrounding. Also a large amount of image data is easily available on the internet so we can always find similar images even if we don't have them right away.

Looking back at the "House of peppercorns" and the "Peppercorns of houses" it seems also compelling to investigate if we could improve the way of computing the similarity of two images to better match the way we humans percieve structures.

If we want the resulting image to consist of real image patches and not a calculated mean of the patches of a cluster, we could try to select the best actual patch from a cluster to represent a group in the dictionary. As mentioned in section 2.4 we also tried the k-medoids clustering algorithm instead of the k-means algorithm. With k-medoids the results were close but not quite as good as with k-means. It seems that the centroids

represent the entire data better than the closest actual representative to the mean. This could be due to the fact that the calculated means compose more precise and clear edge like features that are needed to represent a natural image well. Then again selecting the mean of all patches also smoothens the patches and loses some of the small details of individual patches. In future studies it would be interesting to look into ways of finding real image patches from each cluster that represent the data better than the medoid atoms or even better than the centroids.

Bibliography

[1] Yevgen Matviychuk and Shannon M. Hughes. Exploring the manifold of image patches. In *Proceedings of Bridges 2015: Mathematics, Music, Art, Architecture, Culture*, pages 339–342, Phoenix, Arizona, 2015. Tessellations Publishing. Available online at http://archive.bridgesmathart.org/2015/bridges2015-339.html. 2

[2] Julien Mairal, Francis Bach, and Jean Ponce. Sparse modeling for image and vision processing. *Found. Trends. Comput. Graph. Vis.*, 8(2-3):85–283, December 2014. 2, 7, 17

[3] David Zhang and Zhou Wang. Image information restoration based on long-range correlation, 2002. 4

[4] Sara Soltani, Martin S. Andersen, and Per Christian Hansen. Tomographic image reconstruction using dictionary priors. *CoRR*, abs/1503.01993, 2015. 4, 7

[5] Davood Karimi and Rabab K. Ward. Patch-based models and algorithms for image processing: a review of the basic principles and methods, and their application in computed tomography. *Int J CARS*, 2016. 5, 17

[6] Aapo Hyvärinen, Jarmo Hurri, and Patrick O. Hoyer. *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision.* Springer Publishing Company, Incorporated, 1st edition, 2009. 5, 21

[7] Jean-Luc Starck, Fionn Murtagh, and Jalal Fadili. *Sparse Image and Signal Processing: Wavelets, Curvelets, Morphological Diversity.* Cambridge University Press, 2015. 6, 7, 12

[8] Ron Rubinstein, Alfred M. Bruckstein, and Michael Elad. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, 98(6), 2010. 7

[9] Bruno A Olshausen et al. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996. 7

[10] K. Engan, S. O. Aase, and J. Hakon Husoy. Method of optimal directions for frame design. In *Proceedings of the Acoustics, Speech, and Signal Processing, 1999. On 1999 IEEE International Conference - Volume 05*, ICASSP '99, pages 2443–2446, Washington, DC, USA, 1999. IEEE Computer Society. 8

[11] M. Aharon, M. Elad, and A. Bruckstein. Svdd: An algorithm for designing overcomplete dictionaries for sparse representation. *Trans. Sig. Proc.*, 54(11):4311–4322, November 2006. 8

[12] Adam Coates and Andrew Y. Ng. Learning feature representations with k-means. In *Neural Networks: Tricks of the Trade - Second Edition*, pages 561–580. Springer, 2012. 8, 12, 15

[13] Weisheng Dong, Lei Zhang, Guangming Shi, and Xiaolin Wu. Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization. *Trans. Img. Proc.*, 20(7):1838–1857, July 2011. 10

[14] Weisheng Dong, Lei Zhang, Guangming Shi, and Xin Li. Nonlocally centralized sparse representation for image restoration. *Trans. Img. Proc.*, 22(4):1620–1630, April 2013. 10

[15] Chris Solomon and Toby Breckon. *Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab*. Wiley Publishing, 1st edition, 2011. 11

[16] Michal Aharon. *Overcomplete Dictionaries for Sparse Representation of Signals*. PhD thesis, Technion - Israel Institute of Technology, 2006. 12

[17] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. 15

[18] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *Trans. Img. Proc.*, 13(4):600–612, April 2004. 16

[19] Teemu Helenius and Samuli Siltanen. Photographic dataset: Random peppercorns. *arXiv:1603.01046*, 2016. 19

[20] Júlio César Ferreira, Elif Vural, and Christine Guillemot. Geometry-aware neighborhood search for learning local models for image superresolution. *IEEE Trans. Image Processing*, 25(3):1354–1367, 2016. 22, 23

[21] Gabriel Peyré. Manifold Models for Signals and Images. *Computer Vision and Image Understanding*, 113(2):249–260, February 2009. 23